# Lab 3 Alarm Clock

## Table of Contents

---

## 0 Repository Structure

The typical explanation for the repo structure. Lab specific instructions can be found further below.

### 0.1 HW

The `hw` folder should contain your schematic and board files for your PCB or circuits. In labs 1-5 and 10, you will be creating schematics for your circuit in EAGLE. A setup tutorial can be found here.

### 0.2 SW

The `sw` folder should contain your application firmware and software written for the lab. The SW/inc folder contains firmware drivers written for you by Professor Valvano. Feel free to write your own (in fact, in some labs, you may be required to write your own).

You can place any other source files in the `sw/` folder. TAs will look at the files you create and/or modify for software quality and for running your project.

### 0.3 Resources

A couple files are provided in the Resources folder so you don't have to keep searching for that one TI document. Some of them are immediately useful, like the TM4C datasheet. Others may be useful for your final project, like the TM4C_System_Design_Guidelines page.

### 0.4 Git and Github

We will extensively use Git and Github for managing lab projects. This makes it easier for TAs to grade and help debug the project by allowing us to see commit histories, maintain a common project structure, and Likewise, it makes it easier for students to collaborate with partners, merge different codebases, and to debug their work by having a history of commits.

Two common ways of using Git and Github are Github Desktop and the command line. Tutorials are also abundant on the net for you to peruse. We've provided a cheatsheet for git in the Resources folder.

It is highly recommended to make the most out of Git, even if you've never used it before. Version control will save you a lot of suffering, and tools like Git or SVN are ubiquitous in the industry.

A gitignore file is added to the root of this repo that may prevent specific files from being tagged to the repo. This are typically autogenerated output files we don't care about, but sometimes other stuff (like .lib files) falls through that we want. Feel free to modify if necessary.

---

## 1 Summary

### 1.1 Goals

- Develop a graphics driver for the LCD that can display clock faces
- Design a hardware/software interface for a keyboard or individual switches
- Design a hardware/software driver for generating a simple tone on a speaker
- Measure supply current necessary to run the embedded system
- Implement a digital alarm clock using periodic interrupts

### 1.2 Review

- Valvano Section 3.4 on developing modular software
- Valvano Sections 4.5 and 4.10 on edge-triggered interrupts and the keypad
- Valvano Section 5.7 on periodic interrupts

### 1.3 Team Size

The team size for this lab is 2.

> Two shall be the number thou shalt count, and the number of the counting shall be two. Three shalt thou not count, neither count thou one, excepting that thou then proceed to two. Four is right out.

### 1.4 Background

Labs in ECE445L are extremely open ended. For Labs 3, 4 and 5 you will be given a requirements document. Your TA is your client or customer. A grade of B can be achieved by satisfying the minimum specifications listed by the document. To achieve higher grades, you are expected to expand sections 2.1 and 2.5 describing what your system will do. You are free to make any changes to this document as long you achieve the educational goals for the lab. All changes must be approved by your TA. Excellent grades are reserved for systems with extra features and are easy to operate. You will need your LaunchPad and color LCD. From checkout you can borrow speaker, switches, IRLD024 or IRLD120 MOSFET, 1N914, and some resistors for this lab.

### 1.5 Required Hardware

ECE445L groups should obtain these parts.

| Part | Website | Price |
|---|---|---|
| EK-TM4C123GXL | http://www.ti.com | \$12.99 |
| Sitronix ST7735R Color LCD | http://www.adafruit.com/products/358 | \$12.99 |

### 1.6 Required Parts

These are available at the ECE checkout desk.

| Part | www.mouser.com | www.element14.com | www.digikey.com | Price |
|---|---|---|---|---|
| IRLD120 MOSFET | 844-IRLD120PBF | 19K8369 | IRLD120PBF-ND | $1.07 |
| IRLD024 MOSFET | 844-IRLD024PBF | 19K8367 | IRLD024PBF-ND | $1.12 |
| Tactile switch | 653-B3F-1050 | B3F-1050 | SW405-ND | $0.29 |
| Speaker (8 or 32 ohm) | | | | |

Any 8 to 32 ohm speaker will suffice, search for a speaker at * https://www.mouser.com * https://www.digikey.com * https://www.element14.com

### 1.7 Requirements document

Requirements Document.docx

---

## 2 Preparation

1. Edit the provided requirements document to reflect your proposed design.
2. Draw a detailed circuit diagram showing all external hardware connections. We expect you to use Eagle (because this is the program with which we will be designing PCBs in Labs 6 and 7). More specifically, we want you to use the same CAD program you plan to use in Labs 6 and 7. Label all hardware chips, pin numbers, and resistor values. You do have to show connections to the LaunchPad, but not circuits within the LaunchPad itself. You must have in your possession all external hardware parts, but you do not have to construct the circuit. To limit the surge current into the MOSFET for the speaker circuit, we recommend a 10k resistor between the digital output (500 Hz squarewave) and the gate of the MOSFET.
3. For each module you must have a separate header and code file. As stated earlier we expect at least four modules. As part of the preparation, you need to have the software designed, written and compiled. For the preparation, you do not need to have run or debugged any code.
4. Write one main program that implements the digital alarm clock. Figure 3.1 shows the data flow graph of the alarm clock.

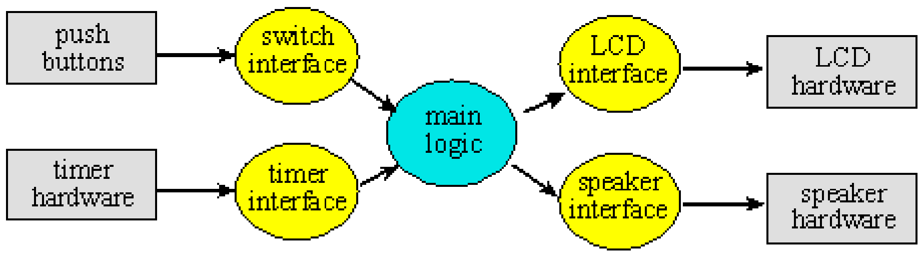


*Figure 3.1. Data flows from the timer and the switches to the LCD and speaker.*

Figure 3.2 shows a possible call graph of the system. Dividing the system into modules allows for concurrent development and eases the reuse of code.

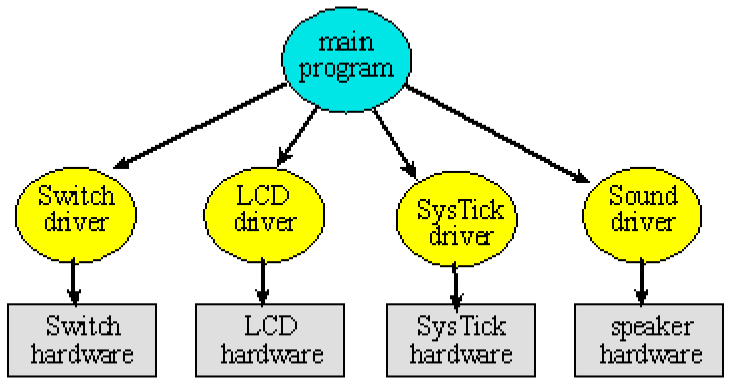


*Figure 3.2. A call graph showing the four modules used by the alarm clock.*

## 3 Procedure

### 3.1 Build Circuit and Critical Sections

1. Build and test any external hardware needed. Debug each module separately. Debug the overall alarm clock. Measure how long it takes to update the graphical time on the LCD. Identify all shared I/O ports and global variables.
    - i.e. document in your software all the permanently allocated variables that have read or write access by more than one thread.
2. Next, consider what would happen if the interrupt occurred between any two instructions of the main program. Remember that high priority interrupts can suspend lower priority ISRs. Look for critical sections, and if you find any, remove them. Document in your software that each shared object is not critical. During checkout, the TA may ask you to prove that your system has no critical sections.

### 3.1.1 Deliverable 1a

Draw the electrical circuit you used to create the alarm clock. You do not need to include any circuits on the LaunchPad (PF4-PF0 circuits).

### 3.1.2 Deliverable 1b

Measure how long it takes the LCD graphics to update.

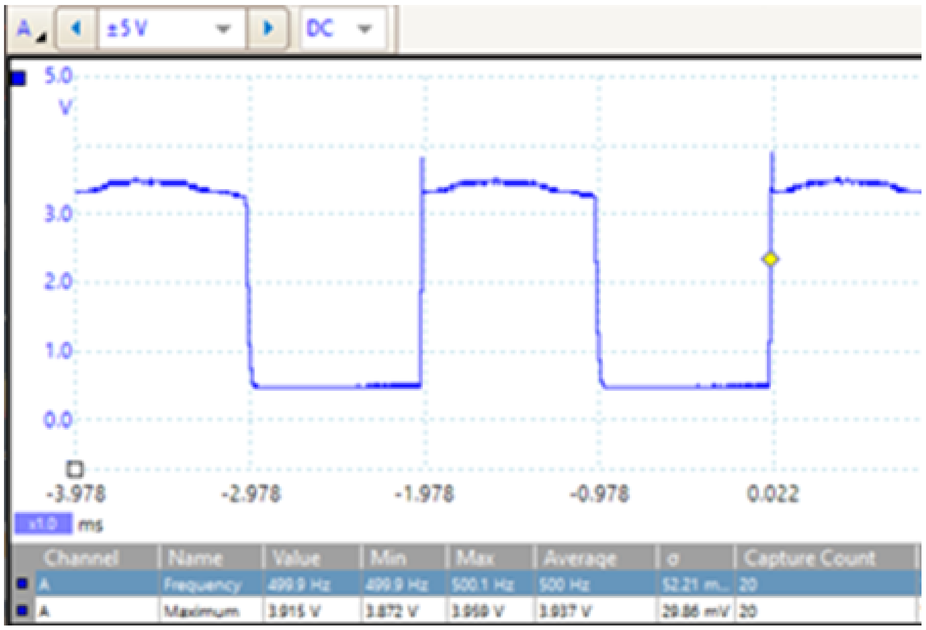
## 3.2 Measure Noise

Record the +3.3V power line rms noise level. The easiest way to measure rms is to use the voltmeter in AC voltage mode. Theoretically one would expect this rms level to be zero, but the presence of noise will cause the rms value to be somewhere in the range of 0.5 to 5 mV.

### 3.2.1 Deliverable 2

Show the RMS noise level on the 3.3V with and without the alarm sounding.

If you have access to a real oscilloscope, use it to measure the speaker voltage when the alarm is sounding. * i.e., measure the drain pin of the MOSFET (bottom side of the speaker). * Notice the small voltage spikes (4V peaks) that occur when current is turned off. Use the scope to verify the sound frequency is 500 Hz.



| Channel | Name | Value | Min | Max | Average | σ | Capture Count |
|---|---|---|---|---|---|---|---|
| A | Frequency | 499.9 Hz | 499.9 Hz | 500.1 Hz | 500 Hz | 52.21 m... | 20 |
| A | Maximum | 3.915 V | 3.872 V | 3.959 V | 3.937 V | 29.86 mV | 20 |

*Figure 3.3a. Scope trace of the voltage on the speaker (the other speaker pin is +3.3V). Current flows when the drain pin is low. 8-ohm speaker 10k resistor*

*between pin and gate, and IRLD120 MOSFET.*

## 3.3 Measure Speaker Voltage

*Note: Do either 3a or 3b, but not both.*

### 3.3.1 Deliverable 3a (real oscilloscope)

If you have access to a real oscilloscope, use it to measure the speaker voltage when the alarm is sounding. I.e., measure the drain pin of the MOSFET (bottom side of the speaker). Notice the small voltage spikes (4V peaks) that occur when current is turned off. Use the scope to verify the sound frequency is 500 Hz.

Measure the voltage versus time of the drain pin of the MOSFET, and use it to determine the current through the speaker. Measure the frequency of the sound. Place a picture of the scope trace like Figure 3.3a into your lab report, either a photo or digital downloaded image.

### 3.3.2 Deliverable 3b (TExaS oscilloscope)

If you do not have access to a real scope, use the TExaS scope to measure the **gate** of the MOSFET (digital squarewave), like Figure 3.3b. Verify the sound frequency is 500 Hz. Because the TExaS scope has a limit of 0 to 3.3V, please do not connect PD3 to the **drain** of the MOSFET.

Use TExaS scope to measure voltage versus time on the gate pin of the MOSFET, and use it to measure the frequency of the sound. Place a screen shot of the scope trace like Figure 3.3b into your lab manual.

## 3.4 Measure Current Draw

Measure the total current drawn by the entire system: 1. Turn off power before connecting or disconnecting. 2. Disconnect the USB cable from the PC. 3. Set the DC power supply limits to +5V and 500mA 4. Connect the DC power supply to the LaunchPad's VBUS and GND 5. Measure the amount of current being drawn from the DC power supply

Measure the current drawn by each device: 1. Use a multimeter to measure the current running to each device separately 2. Devices to measure: * Speaker * LCD screen

### 3.4.1 Deliverable 4

Show the system current and power usage with and without the alarm sounding.

---

## 4 Checkout

You should be able to demonstrate all the required and "cool" features of your digital alarm clock. Demonstrate that your digital alarm clock is stand-alone by turning the power off, then on. The digital alarm clock should run (the time will naturally have to be reprogrammed) without downloading the software each time.

## 5 Lab Report

### 5.1 Deliverables

1. Objectives (final requirements document)
2. Hardware Design
   - Create a schematic or figure showing all external components connected to the TM4C123 board. You do not need to show hardware components on the TM4C123 LaunchPad board.
   - Deliverable 1a
   - Deliverable 1b
3. Software Design (upload your files as instructed by your TA)
   - If you organized the system different than Figure 3.1 and 3.2, then draw its data flow and call graphs
4. Measurement Data
   - Deliverable 2
   - Deliverable 3a
   - Deliverable 3b
   - Deliverable 4

### 5.2 Analysis and Discussion

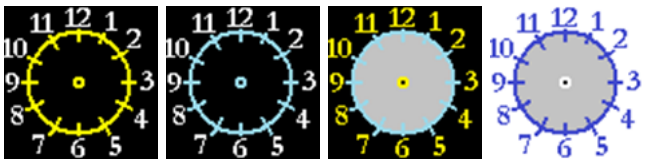
Give short 1 or 2 sentence answers to these questions.

1. Give two ways to remove a critical section.
2. What would be the disadvantage of updating the LCD in the background ISR?
3. Did you redraw the entire clock for each output? If so, how could you have redesigned the LCD update to run much faster, and create a lot less flicker? If not, how did you decide which parts to redraw?
4. Assuming the system were battery powered, list three ways you could have saved power.

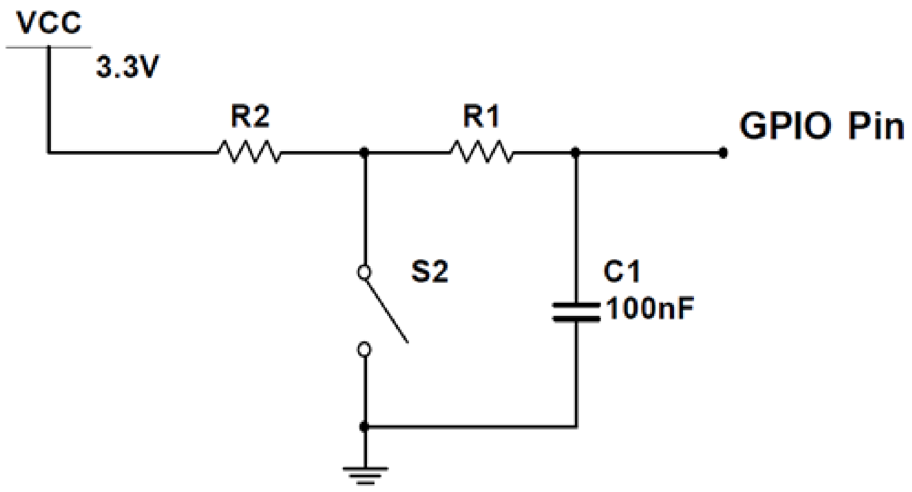
## 6 Hints

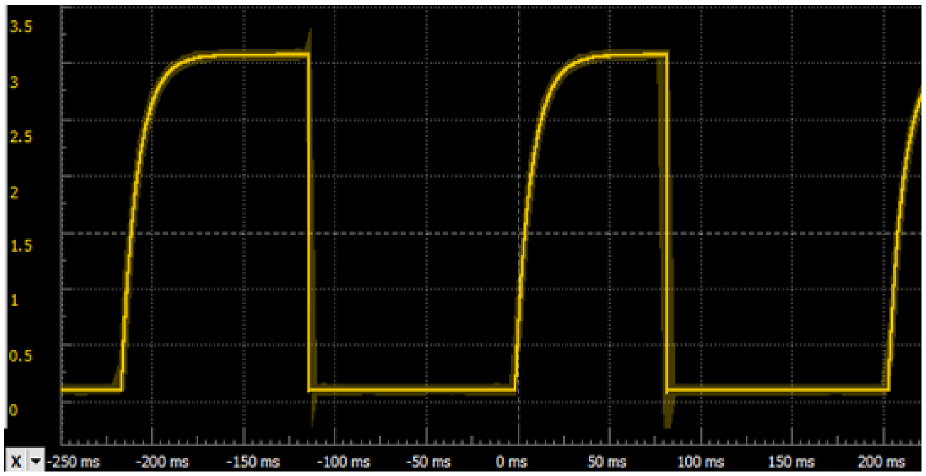
1. The requirements document should change a couple of times during the lab as you determine features that can and cannot be performed.

2. You can interface an 8ohm or 32ohm speaker to an output port using an NPN MOSFET like the IRLD120. A 10k resistor between digital output pin and gate reduces current surges but does not affect loudness. Loudness is determined by the voltage drop across the speaker. Connect the source to ground, and the drain to one side of the speaker. Connect the other side to +3.3V. The maximum IDS of the transistor must be larger than 3.3V / 8ohm or (3.3V / 32ohm). The speaker has inductance, but the MOSFET includes an internal diode to remove back EMF when the transistor switches off. If you toggle the output pin in the background ISR, then sound will be generated. Loudness is determined by the voltage drop across the speaker. From Figure 3.3a, we see the MOSFET drain voltage is about 0.5V when active. So, the voltage drop will be 3.3V-0.5V = 2.7V.
3. You must be careful not to let the LCD show an intermediate time of 1:00 as the time rolls over from 1:59 to 2:00. You must also be careful not to disable interrupts too long (more than one interrupt period), because a time error will result if any interrupts are skipped.
4. You may use 32-bit or 64-bit timer modes on the TM4C microcontrollers. However, it is good practice to refer to the errata for the microcontroller you are using. The errata describe bugs and flaws not listed in the data sheet.
5. If you use the on-board switches then you must activate the internal pull-up resistors. In particular, you will set the corresponding bits in the GPIO_PORTF_PUR_R register. These on-board switches are simply SPST switches to ground. When the switch is pressed, the signal goes to 0V (ground). When the switch is not pressed, the internal pull-up makes the signal go high (3.3V). Furthermore, coming up out of a reset PF0 is locked, and thus if you use PF0 you will need to unlock it.
6. Learn to use Eagle. You will need to be proficient with this application during Labs 6 and 7. Using it now for simple circuits will be an efficient use of your time.
7. One way to design the graphical interface is to draw the static components of the clock in Paint, save as BMP, and convert the BMP into C code, and then use `ST7735_DrawBitmap` to draw the static image on the screen.



8. If you use edge-triggered interrupts, build an analog filter to debounce each switch. Set R1=0, and R2=100k to create the negative logic switch. Choose R2 and C1 so the time constant (tau=R2*C1) is around 10 ms. Test the circuit with a scope before connecting to the microcontroller.





9. A better method to debounce edge-triggered interrupts is to use a second timer (see `EdgeInterruptDebounce_4C123`).